**ProductApp** to aplikacja internetowa umożliwiająca zarządzanie produktami, zaprojektowana jako single-page-application (SPA). Została zbudowana z myślą o spełnieniu wszystkich wymagań funkcjonalnych projektu:

* Stworzona według autorskiego pomysłu,
* Składa się dokładnie z 7 podstron:
  + HomePage – strona główna
  + LoginPage – strona logowania
  + RegisterPage – strona rejestracji
  + ProductsPage – lista produktów
  + EditProduct – formularz edycji produktu
  + ProductDetails – podgląd szczegółów produktu
  + AddProduct – strona dodania nowego produktu
* Zawiera widoki:
  + Lista – lista produktów (ProductsPage)
  + Szczegóły – szczegóły produktu (ProductsDetails)
  + Dodawanie – dodanie produktu (AddProduct)
  + Edycja – edycja produktu (EditProduct)
* Posiada dwa formularze z pełną walidacją – klient, serwer: RegisterPage, AddProduct
* Zaimplementowane zostało uwierzytelnianie w oparciu o technologię Jason Web Token (JWT)

**Frontend** napisany został przy użyciu frameworka **Vue.js**:

* Do wykonywania zapytań HTTP użyto biblioteki Axios
* Komunikacja z serwerem odbywa się z użyciem endpointów JSON
* Do wykonania warstwy prezentacji wykorzystano framework Bootstrap

Backend napisany w języku C# i stworzony jako projekt **ASP .NET Core Web API**:

* Jako system zarządzania bazą danych wykorzystano Microsoft SQL Server
* Do komunikacji serwera z bazą danych w oparciu o mapowanie obiektowo-relacyjne (ORM) użyto technologii Entity Framework Core
* Zaimplementowany mechanizm autoryzacji i uwierzytelniania zapewnia, że endpointy są chronione i wymagają ważnego tokena JWT do uzyskania dostępu

**Endpointy**:

AuthController

* POST /api/Auth/register

Rejestruje nowego użytkownika

Request Body:

{

"username": "string", //string

"password": "string" //string

}

Response:

- 200 OK

- 400 BadRequest – jeśli dane są nieprawidłowe

* POST /api/Auth/login

Loguje użytkownika i zwraca token JWT

Request Body:

{

"username": "string", //string

"password": "string" //string

}

Response:

{

"token": "string" //string

}

- 200 OK

- 401 Unauthorized - jeśli dane logowania są nieprawidłowe

- 400 BadRequest – jeśli dane są nieprawidłowe

ProductsController

* GET /api/Products

Zwraca listę wszystkich produktów. Wymaga autoryzacji

Response:

[

{

"id": 1, //int

"name": "string", //string

"description": "string", //string

"price": 0.0, //double

"quantity": 0 //int

}

]

- 200 OK

- 401 Unauthorized – brak ważnego tokenu JWT

* GET /api/Products/{id}

Zwraca szczegóły produktu o podanym ID w URL. Wymaga autoryzacji

Response:

{

"id": 1, //int

"name": "string", //string

"description": "string", //string

"price": 0.0, // double

"quantity": 0 //int

}

- 200 OK

- 404 Not Found – jeśli taki produkt nie istnieje

- 401 Unauthorized – brak ważnego tokenu JWT

* POST /api/Products

Dodaje nowy produkt. Wymaga autoryzacji

Request Body:

{

"name": "string", //string

"description": "string", //string

"price": 0.0, //double

"quantity": 0 //int

}

Response:

- 201 Created

- 400 Bad Request – jeśli dane są nieprawidłowe

- 401 Unauthorized – brak ważnego tokenu JWT

* PUT /api/Products/{id}

Aktualizuje produkt o podanym ID. Wymaga autoryzacji

Request Body:

{

"id": 1, //int

"name": "string", //string

"description": "string", //string

"price": 0.0, //double

"quantity": 0 //int

}

Response:

- 204 No Content

- 400 Bad Request – jeśli dane są nieprawidłowe

- 404 Not Found – jeśli produkt nie istnieje

- 401 Unauthorized – brak ważnego tokenu JWT

* DELETE /api/Products/{id}

Usuwa produkt o podanym ID w URL. Wymaga autoryzacji

Response:

- 204 No Content

- 404 Not Found – jeśli produkt nie istnieje

- 401 Unauthorized – brak ważnego tokenu JWT

Autor pracy : Krzysztof Szymczyk

Wszystko wykonałem samodzielnie